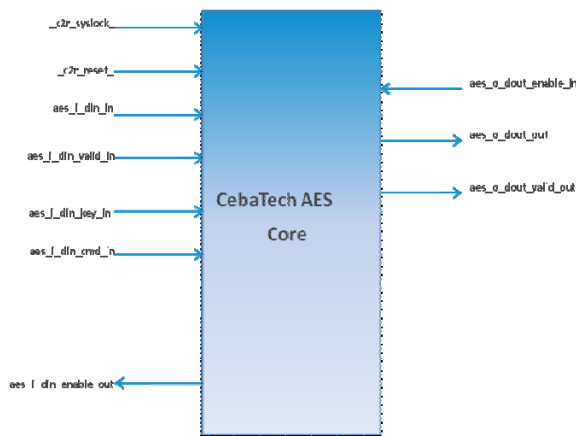


AES Core

Overview

CebaTech's family of Advanced Encryption Standard (AES) IP cores, provide state of the art hardware encryption and decryption to meet data security needs. CebaTech's hardware implementation of AES is in the form of a standalone soft core that implements the Rijndael algorithm as specified by NIST in FIPS PUB 197. AES is a symmetric key block cipher that works by encrypting blocks of 128 bits. The core supports three different key lengths - 128 bits, 192 bits and 256 bits as specified in the standard.

The AES family of cores enable complete FPGA or ASIC solutions for applications that require data security like data storage backup systems, IPSEC and secure networking appliances.



Deliverables

- Synthesizable Verilog™ RTL Source
- Simulation Environment and Scripts
- Verilog test bench with test scripts
- Comprehensive User's Guide
- System C model

CebaTech's AES Features

- Self-contained, standalone soft core suitable for ASIC and FPGA targets
- Fully synchronous design
- Fully compliant with FIPS PUB 197
- Configurable to support AES-128, AES-192, AES-256
- Customizable AES Encrypt or AES Decrypt only cores
- Throughputs in excess of 18 Gbps
- Customizable for Area and Performance requirements of the customer
- Internal Key Generation Block
- Other AES operating modes (ECB, GCM, GMAC and XTS) can be supported
- Easily combined with CebaTech's GZIP and GUNZIP compression and decompression cores for complete storage subsystem solutions

AES Encryption/Decryption

The US Government (NIST) adopted the AES specification in November 2001 as the federal standard for the encryption of sensitive data. The Rijndael algorithm was selected for implementation of the AES specification. AES is a block cipher that works by encrypting groups of 128 message bits into 128 bit encrypted blocks. AES operates on a 4x4 array of bytes (16 bytes or 128 bits) termed the *state*. The AES-128 algorithm consists of 10 steps called *rounds*. For encryption, each round of AES (except the last round) involves the four stages— SubBytes, ShiftRows, MixColumns and AddRoundKey. Each round of AES uses a different key and these round keys are generated from a master cipher key. The key generation algorithm for the key lengths of 128, 192 and 256 bits are also specified in the FIPS PUB 197 standard.

The AES Decrypt operation involves the inverse of the encrypt transformations in the reverse order namely InvShiftRows, InvSubBytes, InvMixColumns and AddRoundKeys. The same key generation algorithm is used and keys are used in the reverse order.

AES Configuration and Performance

Multiple configuration and run-time options for CebaTech's AES core provide for tremendous flexibility to match the core characteristics with the user application. Configuration time options determine how the core is built, and impact performance in terms of throughput and area. These features, along with the option to tile the AES core to scale throughput, enable CebaTech to rapidly deliver a broad range of advanced hardware security solutions. All the configuration options provided in the table can be used for the Encrypt only and Decrypt only blocks also.

Configuration Option	Description
Pipelined Architecture	Enables pipelining of the AES architecture. Pipelined AES provide higher throughput at the cost of increased area
Table-based Implementation	Enables storing the constants in memory blocks. In FPGAs, using Block RAMs to store the constants reduces the usage of logic elements
Memory:Logic Ratio	Ratio of Memory blocks to logic elements used for storing constants. Sum of memory and logic must be 10. Option used only when Pipelined Architecture is enabled.
Key Length	Length of the input master key. Valid values are 128, 192,256.

AES Core

Pipelined Architecture

Enabling the Pipelined Architecture option configures an architecture that pipelines the round operations of the encrypt/decrypt operation. The pipelined architecture provides higher throughput but consumes more area than a non-pipelined architecture as each round is performed concurrently. The pipelined architecture consist of ten pipeline stages and after an initial delay for filling the pipeline, the core produces an output during every clock cycle.

Table based Implementation

The constants that are used in the transformations of the encrypt/decrypt operations can be stored in the memory blocks, thereby reducing the amount of other logic resources that are used. This option is primarily useful in FPGAs, where a number of block RAMs are available. If the constants are not mapped to tables in memory, the round operations are implemented entirely using the available logic resources.

Memory:Logic Ratio

This option is valid only when the Pipelined Architecture option is enabled. In the pipelined architecture, a combination of memory and logic resources can be used to store the constants. The module can be configured to use a combination of memory and logic depending on the number of memory blocks available. Each stage of the ten pipelined architecture can either use memory block or logic resources and hence, the Memory plus logic in the ratio must equal ten. The valid values can range from 10:0 (maximum memory Usage) to 0:10 (no memory usage).

Resource Requirements

CebaTech's AES cores are suitable for ASICs and FPGAs. Resource requirements vary greatly based on the configuration required for a specific application.

Using the C2R development methodology, CebaTech works with each customer to develop application specific IP core requirements and helps customers evaluate the various core configuration options to ensure optimum application-level solutions. Contact sales@cebatech.com to start your customized IP core evaluation today!

This document contains information proprietary to CebaTech Inc. CebaTech retains all intellectual property rights to all products identified in this document. All information is supplied strictly "as is" with no warranties implied or expressed. CebaTech, Inc shall not be liable for any loss or damage arising from the use of any information contained in this document.